Sentiment Analysis on Arabic Reviews

Basant Allam, John Sedrak, Lojain El Sayed

June 2, 2023

1 Introduction

Sentiment analysis is a rapidly growing field of research that aims to automatically extract and classify opinions, attitudes, and emotions expressed in text data. With the increase and spread of digitized information and social media, sentiment analysis has become a vital tool for businesses, governments, and researchers to collect opinions on products, services, and events. However, most of the existing sentiment analysis research has focused on English and other widely spoken languages, leaving a gap in the analysis of sentiment in languages with a smaller digital presence. Arabic, being the fifth most spoken language in the world, is one of the languages that has received much less attention in the field of sentiment analysis.

This report aims to explore sentiment analysis on a set of hotel, book, and airline reviews and classify them as positive, negative, or mixed while tackling challenges such as spelling mistakes and lack of diacritic usage. The report also aims to highlight the challenges of performing sentiment analysis on Arabic text.

2 Related Work

Although a lot of previous work was done on fine-tuning MARBERT for sentiment analysis tasks [12], [11], as far as we know, MARBERT was never fine-tuned on the 100K Reviews dataset, thus we thought it would be a valuable contribution to experiment with this model and dataset.

3 Challenges and Approach

Performing sentiment analysis on any data collected from the internet introduces several challenges such as URLs, hashtags, misspelled words, slang words, and words with repeated letters (eg. Wooow amazinggg!!). For Arabic text specifically, there is the added issue of handling the lack of diacritics known as "Tashkeel" (تشكيل) and the lower availability of data in comparison with English.

3.1 Misspellings and Slang

Misspellings, slang, and words with repeated letters are by far the most common challenge that is faced when performing any sort of natural language processing on internet-collected data, such as reviews and tweets, because there are no regulations or restrictions that require maintaining proper grammar or spelling.

A reasonable approach would be to process all the words and remove letters that appear more than two times in succession (ie. thrice in a row). We allow for double letters to include words such as "color" (اللون). We can then use a spellchecker to find the closest candidate for every word, which will eliminate (to a good extent) misspellings and any invalid double letters. Our approach for slang words will be leaving them as is and allow the fine tuning to learn their meanings.

3.2 URLs and Hashtags

URLs and hashtags are a natural consequence of collecting any web based data. In the context of sentiment analysis, the content of URL strings is seldom useful. Hashtags are usually several words

concatenated into a single word, which makes them gibberish to the sentiment analysis model, so they should not be included as well.

A reasonable approach that is often done is to replace all URLs and hashtags with a [URL] token and a [HASHTAG] token respectively. This way, we remove gibberish while maintaining context in the corpus. However, our corpus did not contain any URLs or hashtags, rendering the suggested approach pointless.

3.3 Diacritics

Figure 1: A sample Arabic text. Please note that Harakat is another word for Tashkeel [3].

Arabic is one of the four abjad languages. Abjad languages have scripts that only write the consonants and long vowels as letters, leaving the short vowels to be handled by diacritics (as shown in Figure 1). In Arabic, these diacritics are known as tashkeel (تشكيل). For this reason, many words have the same spelling, but differ in their diacritics. Usually, native Arabic speakers can easily infer the diacritics on a word given its context, and thus infer the meaning of the word itself. For this reason, diacritics are usually omitted from text, making it difficult for computers to determine which word embedding should be used when.

We are considering two potential approaches for dealing with diacritics:

1. Using diacritic restoration tools to estimate diacritics from contexts.
2. Removing diacritics from all words to somewhat simplify the task.

As our corpus contained no words with diacritics, we followed approach 2.

3.4 Availability

While Arabic is a very rich language with numerous pieces of literature and poetry, the amount of user-generated Arabic content on the internet is very limited. Furthermore, spoken Arabic and formal, written Arabic are widely different, so models cannot be solely trained on written Arabic then applied directly to casual spoken Arabic. When dealing with internet reviews and tweets, most people write the same way they would speak. Therefore, in order to perform sentiment analysis on internet content, models should be trained – or at the very least fine-tuned – on internet content.

To overcome the challenge of limited data availability, we will be using a pre-trained model and fine-tuning it on our dataset.

4 Dataset

The dataset we will be using is 100K Arabic Reviews Dataset[2]. This is an Arabic Dataset consisting of 99,999 reviews written in Arabic and it is mainly used in Sentiment Analysis tasks. The reviews are about hotels, books and airlines. The Hotel and book reviews from the HARD[4] and BRAD [1] datasets respectively, while the airline reviews were collected manually from 100 airlines reviews. The dataset has 2 fields: the text and its corresponding label being either positive, negative or mixed.

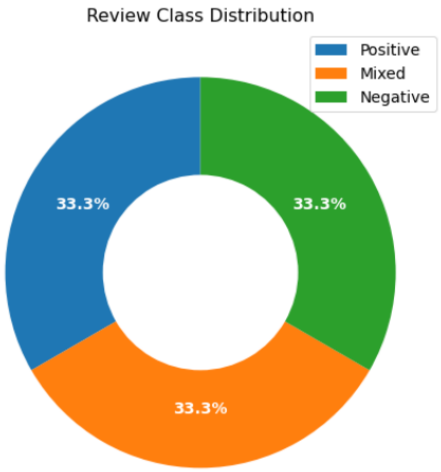
Class	Sample Data Points
Positive	ممتاز . كل شي ممتاز في الفندق طاقم الاستقبال رائع والفندق رائع ونظيف.
Mixed	مرضي. سعر الغرف غالي جدا
Negative	سوء أدب موظف الاستقبال مخيب للأمل.

Figure 2: Sample Data from 100K Reviews Dataset. [2].

4.1 Data Analysis

In order to better understand the dataset, we did some data analysis to find certain aspects of the dataset such as class distribution, length, spelling mistakes.

4.1.1 Class Distribution



Class	Data Points
Positive	33,333
Negative	33,333
Mixed	33,333

Figure 3: Class Distribution of Dataset.

The data distribution is balanced having an equal number of datapoints in each of the three classes, which is promising as it lowers the risk of bias while training the model. As shown in Figure 3.

4.1.2 Length of Reviews

Although it is generally more common that people write more when writing a negative review, compared to a positive review, this is not the case with our dataset. We found that the word count of reviews in all three classes are almost the same with negligible differences, as shown in Figure 4. Moreover, for all 3 classes the mean character counts are also very close as shown in Figure 4.

4.1.3 Spelling Mistakes

Since it is common to find typos on the web, we thought it would be useful to find out the number of misspelled words in our dataset. The unique misspelled words turned out to be over 50 percent of the words (65 percent) As shown in Figure 5. This may be due to the fact that people can over-exaggerate

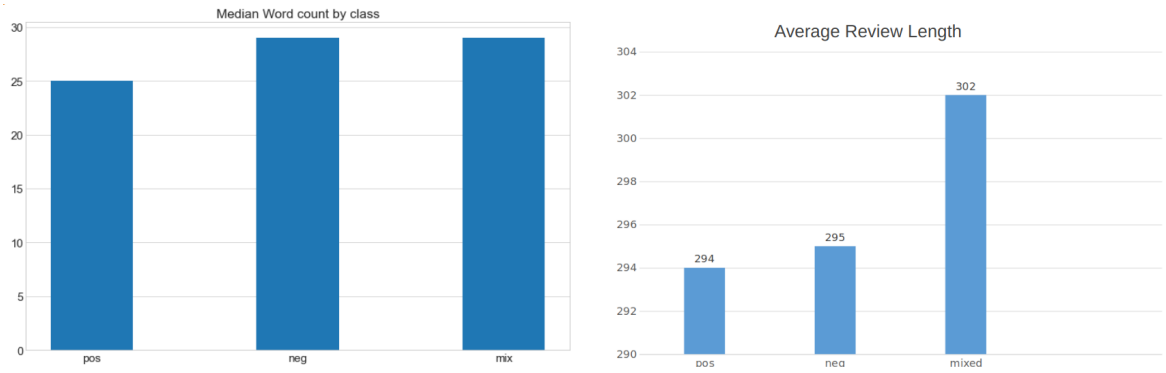


Figure 4: The left figure shows the median word count of every review class. The right figure shows mean review length in terms of number of characters.

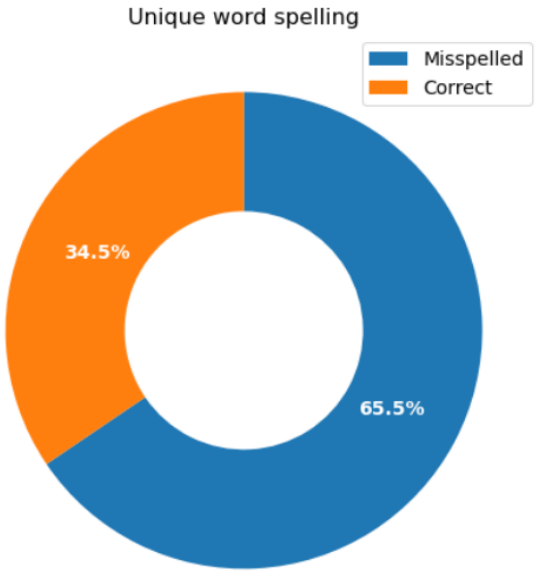


Figure 5: The number of unique words that were misspelled before preprocessing.

by repeating letters resulting in linguistically incorrect words. Thus, we are considering eliminating misspelled words from our dataset so as not to confuse the model.

5 Pre-processing

We decided not to perform lemmatization as the model we will use is BERT so lemmatization is unnecessary. Although the authors of the model we will be using recommended removing diacritics, URLs and Hashtags and implemented this in their own experiments, we found that the dataset we will be using has no diacritics, URLs nor hashtags. As a result, there was no need to do the pre-processing steps recommended by the authors. Instead, we thought of the steps listed below. The first 2 steps were performed in some experiments but not all of them based on the goal of the experiment.

1. Removing Stop Words
2. Spell-Check
3. Padding
4. Adding [CLS] and [SEP] tokens for BERT
5. Tokenization

5.1 Stop Word Removal

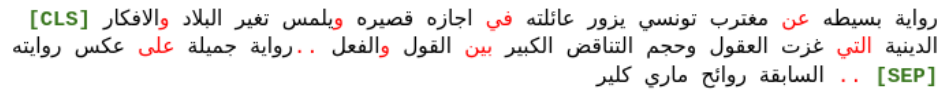


Figure 6: An image highlighting the stopwords that were removed from a single text sample.

For the stop words removal, we used the approach recommended by Arabic Stop Words [13]. However, we did not remove negation words (لا ، ليس) as they are crucial in determining the sentiment of each sentence.

5.2 Spell-Check

We came up with an interesting pre-processing idea that we wanted to experiment with and observe its effect on the model performance. The first stage was to access the entire dataset and remove instances of letters that were repeated more than twice. We then performed a spell-check on every word in the dataset against a pre-defined arabic dictionary which was provided by pyspellchecker package [7] and replaced the misspelled words with the correction suggested by pyspellchecker. This will help remove typos that may confuse the model and affect its performance.

5.3 Padding

If the sentence is less than the fixed length we specify, we add [PAD] tokens to the sentence to unify the length of all sentences fed to the model, since BERT requires a fixed number of tokens.

5.4 Adding CLS and SEP tokens for BERT

We preceded every sentence by [CLS] and ended it with [SEP] in order for Bert to separate each sentence, as recommended by the authors of BERT [10].

5.5 Tokenization

We decided to use the regular BertTokenizer, as recommended by the MARBERT model authors. The BERT tokenizer converted the tokens to their index numbers in the MARBERT vocabulary

6 Model

MARBERT[8] is an Arabic-specific Transformer Language Model pre-trained on a very large and diverse dataset to facilitate transfer learning on MSA as well as Arabic dialects.

6.1 Why MARBERT?

We chose to use MARBERT because it is pretrained on arabic text and thus it will be more familiar with it when we fine-tune it on our dataset compared to using a model pre-trained on another language. Furthermore, MARBERT was shown to be SOTA in several downstream tasks, so we thought it is a promising model to use and the pre-trained model was publicly available as well as their paper.

6.2 Architecture

MARBERT has the same architecture as BERT [10] and uses the BERTBase configuration. As mentioned, we use the same network architecture as BERTBase : 12 layers, 768 hidden units, 12 heads, for a total of approximately 163M parameters.

6.3 MARBERT Pre-training

MARBERT was pre-trained on 1B Arabic tweets randomly sampled from a large in-house dataset of about 6B tweets. Only tweets with at least three Arabic words were included, based on character string matching, regardless whether the tweet has non-Arabic string or not. The dataset makes up 128GB of text which is equivalent to 15.6B tokens.

6.4 MARBERT Fine-tuning

For the fine-tuning, we chose to use the 100K Reviews Dataset mentioned earlier. We shuffled and split the dataset into a train set and a test set by a 80:20 ratio respectively. We conducted 4 different experiments. Then we tested the model and evaluated its performance as we will see in the later sections.

As for the loss function, we used in training, we experimented with two loss functions: Cross Entropy and Multi-label Soft Margin Loss in order to proceed with the better one.

We chose the fine-tuning configuration parameters shown in Figure 7 based on the recommendation of the MARBERT paper [8] authors.

Parameter	Value
Number of Epochs	2
Learning Rate	2e-06
Maximum Sequence Length	128 tokens
Batch Size	32

Figure 7: Fine-tuning parameters

7 Experiments

We conducted 4 experiments. In each experiment we only changed one variable while controlling the other variables in order to accurately determine its effect. For the control and spell-checked experiment, we fine tuned for 5 epochs. After finding that more epochs did not necessarily change the evaluation metrics significantly, we opted to fine tune the multi-label soft margin loss function experiment and the stopword-removal experiment for only two epochs in the interest of time.

In each experiment, we evaluated the performance of the model using the following evaluation metrics provided by sklearn [6]:

1. Training Loss
2. Validation Accuracy
3. Validation Recall
4. Validation Precision
5. Validation F1 Score

7.1 Control Experiment

It is good practice to have a baseline to compare the rest of our experiments to, thus the goal of this experiment was to do minimal pre-processing and compare the rest of the experiments to this one.

epoch_num	train_loss	val_acc	val_recall	val_precision	val_f1	lr
1	0.726922	0.73035	0.730780	0.729674	0.730171	0.000002
2	0.579407	0.73640	0.736757	0.737491	0.737082	0.000002
3	0.539269	0.73830	0.738669	0.738982	0.738793	0.000002
4	0.510306	0.73945	0.739751	0.740842	0.739712	0.000002
5	0.492171	0.73860	0.739036	0.737268	0.737666	0.000002

Figure 8: Table of evaluation metrics for 5 epochs of fine tuning on the raw data, sorted by epoch.

In the first experiment, we trained the model on the pre-processed data without performing neither spellchecking nor stop-word removal. We did this because although on one hand, these pre-processing steps may eliminate words that are not of great effect on the sentiment, on the other hand, they may cause the average number of words per sentence to drop causing insufficient training data, thus we chose to try with and without these 2 steps. In addition, we used the Cross Entropy Loss function.

As can be seen in Figure 8, all the evaluation metrics were fairly unchanged throughout the epochs, with only minor improvements between epoch 1 and epoch 5. Our highest accuracy was 73.9%, and our highest f1 score was 74%

7.2 Spell-Checked Data Experiment

epoch_num	train_loss	val_acc	val_recall	val_precision	val_f1	lr
1	0.722939	0.73055	0.730935	0.730665	0.730780	0.000002
2	0.578857	0.73750	0.737840	0.738650	0.738193	0.000002
3	0.539652	0.73760	0.737885	0.739714	0.738508	0.000002
4	0.510117	0.73645	0.736740	0.738302	0.736791	0.000002
5	0.490089	0.73535	0.735781	0.734194	0.734503	0.000002

Figure 9: Table of evaluation metrics for 5 epochs of fine tuning on the spell-checked data, sorted by epoch.

In this experiment, we fine tuned on data that has been spell-checked according to Section 5.2

Similarly to the control experiment, the metrics did not change significantly between epochs 1 and 5. The maximum accuracy and f1 scores (73.8% and 73.9% respectively) were slightly worse than those in the control experiment. One explanation for this could be that there were words that were grossly misspelled, causing the spellchecker to correct them to known words with very different meanings than the intended words.

7.3 Multi-Label Soft Margin Loss Function Experiment

epoch_num	train_loss	val_acc	val_recall	val_precision	val_f1	lr
1	0.728767	0.7311	0.731578	0.729108	0.729957	0.000002
2	0.578710	0.7362	0.736567	0.736714	0.736587	0.000002

Figure 10: Table of evaluation metrics for 2 epochs of fine tuning on the raw data while using the multi-label soft margin loss function, sorted by epoch.

For this experiment, we tried a different loss function, namely Multi-label Soft Margin Loss [5], rather than Cross Entropy Loss which was used in the control experiment. We did this because it is generally a good practice to try different loss functions and study their effect on the model performance and proceed with the one that yields better results. In the interest of time, the fine tuning was run for only two epochs.

The highest accuracy and f1 scores (73.7% and 73.7% respectively) are not significantly different from those of the previous two experiments.

7.4 Stop-Word Removal Experiment

epoch_num	train_loss	val_acc	val_recall	val_precision	val_f1	lr
1	0.725805	0.709071	0.709398	0.709856	0.709567	0.000002
2	0.616059	0.710721	0.711175	0.708451	0.708701	0.000002

Figure 11: Table of evaluation metrics for 2 epochs of fine tuning on the spell-checked and stopword-free data, sorted by epoch.

In the fourth experiment, we removed stop words – from the spell-checked data – other than negation words (لا، ليس), and studied the effect of their removal on the performance.

As can be seen in Figure 11, the results were significantly worse than the three previous experiments.

8 Limitations

One major limitation of this dataset is that it was not professionally labeled. The labels were determined by the star ratings that were left next to the review, with a rating of less than 3/5 being negative, 3/5 being mixed, and more than 3/5 being positive. This method of labeling is prone to sarcasm and discrepancies between how the reviewer was feeling and how the reviewer expressed those feelings in their review. Furthermore, this method of labeling does not account for the blurred lines that are 2/5 star and 4/5 star reviews.

Unfortunately, this is a limitation that is to be expected from any data collected from the internet, and it is only really avoidable by spending a large amount of resources to hire teams that manually label the data.

Another limitation is working with a "mixed" class of reviews, where reviews could easily be interpreted as positive or negative instead. An example of this could be "This hotel could have been better if it had xyz." In this example, nothing negative was said per se, but the review was mixed nonetheless.

Additionally, any dataset suffers from sarcasm, such as in "عن جد هالكومة عاطينا حريتنا بالكاااااامل هههههههه" [9] (Seriously, this government gives us complete freedom hahahahah). Some of the reviews also lacked context, which made it so the review can be taken as positive or negative, depending on what the review was referring to. This is shown in

"أشرف أجمد واحلى ألف مرة فى الشعر العامى عن الكتابة الساخرة على عكس عمر طاهر" (Ashraf is much better at writing colloquial poetry than satirical writings, as opposed to Amr Taher)

Here, even a human being would struggle to classify this as positive or negative without the relevant context.

9 Conclusion

MARBERT performed decently when fine-tuned on the 100k Arabic Reviews dataset, achieving a peak accuracy of 73.9%. We had hoped for at least 80%, and attempted several preprocessing methods and two different loss functions to achieve that. However, it seems that these different attempts were

fruitless, as the metrics remained nearly the same regardless of the preprocessing applied or loss function used.

This may be a limitation of the dataset as discussed above, or it may be a limitation with the nature of reviews and having a mixed class for them.

10 Future Work

We recommend exploring a sort of tagging system, where reviews can be tagged with class names if the probability of the classes exceeds a certain threshold. This may be helpful for handling the "mixed" class. Using the probabilities of those tags may yet provide insight into the semantic of the reviews without completely misclassifying them.

As for handling exaggerated words such as "جمييييل", after applying our correction algorithm, we can add an exaggeration token such as [VRY] before the corrected word. This can maintain the importance of the word while also bringing it to a form that the model knows.

To handle mislabelled data, a crowd-sourced labelling method can be explored. This method works similarly to existing CAPTCHA methods. It will provide a user with two pieces of text. The sentiment of one of those pieces will be known, and the other will not. The user will be asked to classify the sentiment of both of the texts. If the user classifies the known text correctly, then their classification of the second text will be counted. The unknown text will be presented to multiple users, and if most of them agree on the same sentiment, then the previously unknown text will now be added to the dataset labelled with the agreed upon sentiment.

References

[1] Brad. https://github.com/elnagara/BRAD-Arabic-Dataset.

[2] Dataset 100k arabic reviews. https://www.kaggle.com/datasets/abedkhooli/arabic-100k-reviews.

[3] Elements of arabic script. https://commons.wikimedia.org/wiki/File:Elements_of_Arabic_script_improved.png.

[4] Hard. https://github.com/elnagara/HARD-Arabic-Dataset.

[5] Multilabelsoftmarginloss. https://pytorch.org/docs/stable/generated/torch.nn.MultiLabelSoftMarginLoss.html.

[6] Sklearn Evaluation Metrics. https://scikit-learn.org/stable/modules/model_evaluation.html.

[7] Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. in proceedings of the 10th international conference on language resources and evaluation, 2016.

[8] Muhammad Abdul-Mageed, AbdelRahim Elmadany, and El Moatez Billah Nagoudi. Arbert marbert: Deep bidirectional transformers for arabic, 2021.

[9] Nawaf Abdulla, Nizar A. Ahmed, Mohammed Shehab, Mahmoud Al-Ayyoub, Mohammed Al-Kabi, and Saleh Al-Rifai. Towards improving the lexicon-based approach for arabic sentiment analysis. *International Journal of Information Technology and Web Engineering*, 9:55–71, 07 2014.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[11] Ikram El Karfi and Sanaa El Fkihi. An ensemble of arabic transformer-based models for arabic sentiment analysis. *International Journal of Advanced Computer Science and Applications*, 13(8), 2022.

[12] Abdullah Salem Khered, Ingy Yasser Hassan Abdou Abdelhalim, and Riza Batista-Navarro. Building an ensemble of transformer models for Arabic dialect classification and sentiment analysis. In *Proceedings of the The Seventh Arabic Natural Language Processing Workshop (WANLP)*, pages 479–484, Abu Dhabi, United Arab Emirates (Hybrid), December 2022. Association for Computational Linguistics.

[13] Taha Zerrouki. Arabic Stop Words, 2010.